A new Architecture for OGSA-DAI

Atkinson, M., Antonioletti, M., Baxter, R., Borley, A., Chue Hong, N., Hume, A., Jackson, M., Karasavvas, K., Krause, A., Laws, S., Paton, N., Schopf, J., Sudgen, T., Tourlas, K. and Watson, P.

Introduction

OGSA-DAI is a widely used middleware infrastructure that enables client applications to submit request documents in order to perform a set of activities on a remote data resource (DR) [1]. The initial versions of OGSA-DAI were built using the OGSI infrastructure [2] provided by GT3 [3]. More recently OGSA-DAI has been released on the OMII1 infrastructure [4], based on WS-I+ [5] and on GT4 [6] based on WSRF [7]. OGSA-DAI provides activities to access relational and XML databases, and indexed files. It also provides data translation and third-party delivery activities. This activity framework provides extensibility: application developers can add their own activities. OGSA-DAI will provide a reference implementation of WS-DAI [8]. This paper describes the new architecture and presents its rationale.

Experience

OGSA-DAI has over 1000 registered users [9] including projects which require continuously available data access and integration services. The architecture of the early versions of OGSA-DAI was based on the assumption that web services were light weight. For example, it assumed that new web services could be launched to handle each session between a data resource and a client. Similarly, it assumed that a new service could be produced to handle each data transfer to a data consumer. As a result OGSA-DAI relied on the factory pattern to create independent grid services with a one-to-one relationship with the data they were handling. This meant that grid service handles provided a data naming system, failures were localised to a particular session and that persistence was only required for the accessed and handled data.

New Requirements

The change from OGSI to WS-I+ and WSRF demands a change in the architecture as web services must now be considered long-running. This change has been taken as an opportunity to review all aspects of the OGSA-DAI, taking into account performance and operational issues uncovered through extensive use. The following architectural requirements were considered:

- All of the existing activities and facilities should continue to be supported.
- Each data service must support zero or more data resources (DRs) – dynamic configuration may add and remove DR.
- This exposes complex naming requirements to name DR and use name systems provided by DR.
- Each data service should be resilient to failures within activities and the data resources they access.
- Each data service should be configurable on installation and reconfigurable during operation.
- Each data service should be manageable and persistent.
- The same architecture should accommodate rapid requests for small tasks (execution of an activity) and tasks involving large volumes of or continuous streams of data.
- The data services should support concurrent sessions and transactions.
- The power of request documents should be extended.
- The extensibility framework based on activities should be stabilised to protect application developer investment.

This architecture must be amenable to implementation over a number of platforms and must be achieved incrementally without undue disruption of the development process. The further development of data integration activities must also continue.

Components and Principles

The primary components of the new architecture are shown in Figure 1. The *Data Resource Access Manger* (DRAM) supports data service (re)configuration, monitoring, management and recovery. The initial configuration of a data service is described in the *Data Service Description Language* (DSDL). *Data Services* (DS) have at their core the *OGSA-DAI Engine* (ODE) that provides a framework for activity and task management. A *task* is the execution of an activity on some specified data with specified parameters.

A *Task And Data Document* (TADD) permits a number of tasks to be specified, together with their parameters, inputs, outputs and control flow. This composite format can be used to submit simple requests, to avoid unnecessary round-trip latencies, to report status and results, and to delegate or cascade work. *Data Identifiers* (DIDs) are used to describe the handling of intermediary and externally available results.

The response to a request is generated by the ODE within a *Session*, which may also be a *Transaction* (Tx). The ODE analyses incoming TADDs, conducts authentication and authorisation, and then constructs an optimised execution graph. This is evaluated by spawning activity threads which use pipes and a *Transfer Assembler* pattern to handle arbitrary scale intermediates and results respectively.

The evaluation is overseen by an internal monitoring system to detect excess resource consumption and provide resilience. A DRAM will monitor a DS's status digest produced by its internal monitor. If this shows potential problems it may reconfigure or restart the DS from its last checkpoint. The ODE is being designed to support dynamic configuration, sessions, transactions, recovery and concurrency.

Many standard web service interfaces require specific message protocols; e.g., the protocols to coordinate distributed transactions. They will be implemented by binding their ports to code that constructs an equivalent execution graph. The results produced as a TADD by the transfer assembler are will be translated to the expected result format. As a matter of principle such functions should also be available through a standard activity in a TADD. Conversely, activities that workflow enactments may require should also appear via "standard" WSDL-defined ports. In this way, functions required by one community via a special interface, e.g. WSRF getResourceProperties, will also be available as a task in a request document.

The architecture looks forward to multiple data services administered through a consistent regime. In the Figure one serves OGSA-DAI, one serves the WS-DAI standard [8] perhaps as a configuration of OGSA-DAI and one serves Mobius [10].

Status

The new architecture is currently being prototyped and will be substantially implemented by AHM'05. The paper will fully describe the new architecture and its rationale. It will report experience from implementation and early use.

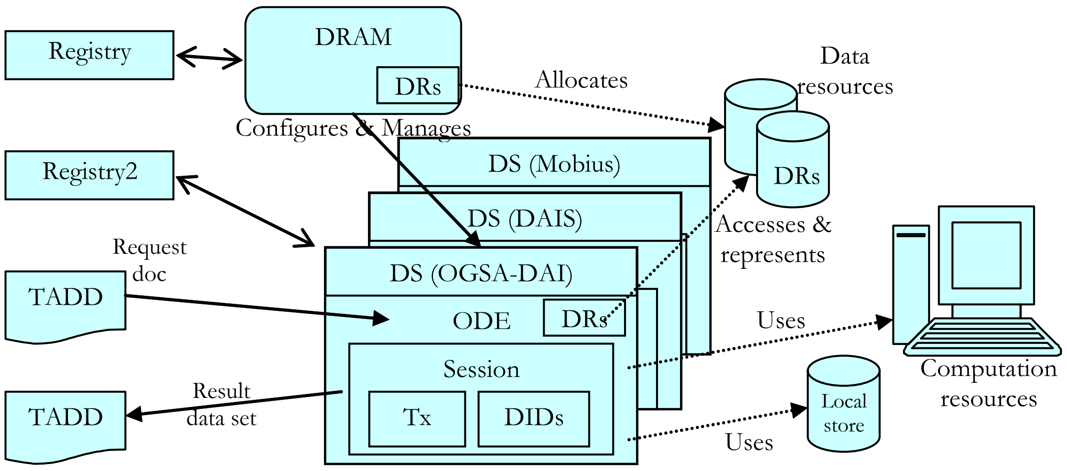

Figure 1: New OGSA-DAI Architecture

References

1. Antonioletti, M., Atkinson, M., Baxter, R., Borley, A., Chue Hong, N., Collins, B., Hardman, N., Hume, A., Knox, A., Jackson, M., Krause, A., Laws, S., Magowan, J., Paton, N., Pearson, D., Sugden, T., Watson, P. and Westhead, M., *The design and implementation of Grid database services in OGSA-DAI.* Concurrency and Computation: Practice and Experience, 2005. **17**(2): p. 357-376.
2. Tuecke, S., Czajkowski, K., Foster, I., Frey, J., Graham, S., Kesselman, C., Maquire, T., Sandholm, T., Snelling and Vanderpilt, D., P., *Open Grid Services Infrastructure, Version 1.0.* 2003, Global Grid Forum.
3. Globus, *Globus Toolkit 3.2.1.* 2004.
4. OMII, *OMII_1.* 2005.
5. Atkinson, M., DeRoure, D., Dunlop, A., Fox, G., Henderson, P., Hey, T., Paton, N., Newhouse, S., Parastatidis, S., Trefethen, A., Watson, P. and Webber, J., *Web Service Grids: an evolutionary approach.* Concurrency and Computation: Practice and Experience, 2005. **17**(2): p. 377-390.
6. Globus, *Status and Plans for the Globus Toolkit 4.0 (GT4) as of February 25 2005.* 2005.
7. OASIS, *WS-ResourceFramework.* 2005.
8. Antonioletti, M., Atkinson, M., Laws, S., Malaika, S., Paton, N. W. , Pearson D., and Riccardi, G. *Web Services Data Access and Integration (WS-DAI).* in *13th Global Grid Forum.* 2005. Seoul, Korea: GGF.
9. Antonioletti, M., Atkinson, M., Baxter, R., Borley, A., Chue Hong, N., Dantressangle, P., Hume, A., Jackson, M., Krause, A., Laws, S., Parsons, P., Paton, N., Schopf, J., Sugden, T., Watson, P. and Vyvyan, D. *OGSA-DAI Status and Benchmarks.* in *(submitted to) UK e-Science All Hands Meeting.* 2005. Nottingham, England.
10. Hastings, S., Langella, S., Oster, S. and Saltz, J. *Distributed Data Management and Integration Framework: The Mobius Project.* in *Proceedings of GGF11 Semantic Grid Applications Workshop.* 2004. Berlin, Germany: GGF.